Homework5

April 19, 2023

1 ECGGR HW5

Patrick Flynn | 801055057

Set everything up:

[44]:
```
!pip install torch torchvision
!pip install d2l==1.0.0a1.post0
!pip install matplotlib_inline
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-
wheels/public/simple/
Requirement already satisfied: torch in /usr/local/lib/python3.9/dist-packages
(2.0.0+cu118)
Requirement already satisfied: torchvision in /usr/local/lib/python3.9/dist-
packages (0.15.1+cu118)
Requirement already satisfied: filelock in /usr/local/lib/python3.9/dist-
packages (from torch) (3.11.0)
Requirement already satisfied: networkx in /usr/local/lib/python3.9/dist-
packages (from torch) (3.1)
Requirement already satisfied: typing-extensions in
/usr/local/lib/python3.9/dist-packages (from torch) (4.5.0)
Requirement already satisfied: sympy in /usr/local/lib/python3.9/dist-packages
(from torch) (1.11.1)
Requirement already satisfied: triton==2.0.0 in /usr/local/lib/python3.9/dist-
packages (from torch) (2.0.0)
Requirement already satisfied: jinja2 in /usr/local/lib/python3.9/dist-packages
(from torch) (3.1.2)
Requirement already satisfied: lit in /usr/local/lib/python3.9/dist-packages
(from triton==2.0.0->torch) (16.0.1)
Requirement already satisfied: cmake in /usr/local/lib/python3.9/dist-packages
(from triton==2.0.0->torch) (3.25.2)
Requirement already satisfied: requests in /usr/local/lib/python3.9/dist-
packages (from torchvision) (2.27.1)
Requirement already satisfied: pillow!=8.3.*,>=5.3.0 in
/usr/local/lib/python3.9/dist-packages (from torchvision) (8.4.0)
Requirement already satisfied: numpy in /usr/local/lib/python3.9/dist-packages
(from torchvision) (1.22.4)
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.9/dist-
```

```
packages (from jinja2->torch) (2.1.2)
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.9/dist-packages (from requests->torchvision) (2022.12.7)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.9/dist-
packages (from requests->torchvision) (3.4)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in
/usr/local/lib/python3.9/dist-packages (from requests->torchvision) (1.26.15)
Requirement already satisfied: charset-normalizer~=2.0.0 in
/usr/local/lib/python3.9/dist-packages (from requests->torchvision) (2.0.12)
Requirement already satisfied: mpmath>=0.19 in /usr/local/lib/python3.9/dist-
packages (from sympy->torch) (1.3.0)
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-
wheels/public/simple/
Requirement already satisfied: d2l==1.0.0a1.post0 in
/usr/local/lib/python3.9/dist-packages (1.0.0a1.post0)
Requirement already satisfied: matplotlib-inline in
/usr/local/lib/python3.9/dist-packages (from d2l==1.0.0a1.post0) (0.1.6)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.9/dist-
packages (from d2l==1.0.0a1.post0) (3.7.1)
Requirement already satisfied: numpy in /usr/local/lib/python3.9/dist-packages
(from d2l==1.0.0a1.post0) (1.22.4)
Requirement already satisfied: gym in /usr/local/lib/python3.9/dist-packages
(from d2l==1.0.0a1.post0) (0.25.2)
Requirement already satisfied: jupyter in /usr/local/lib/python3.9/dist-packages
(from d2l==1.0.0a1.post0) (1.0.0)
Requirement already satisfied: pandas in /usr/local/lib/python3.9/dist-packages
(from d2l==1.0.0a1.post0) (1.5.3)
Requirement already satisfied: requests in /usr/local/lib/python3.9/dist-
packages (from d2l==1.0.0a1.post0) (2.27.1)
Requirement already satisfied: importlib-metadata>=4.8.0 in
/usr/local/lib/python3.9/dist-packages (from gym->d2l==1.0.0a1.post0) (6.3.0)
Requirement already satisfied: cloudpickle>=1.2.0 in
/usr/local/lib/python3.9/dist-packages (from gym->d2l==1.0.0a1.post0) (2.2.1)
Requirement already satisfied: gym-notices>=0.0.4 in
/usr/local/lib/python3.9/dist-packages (from gym->d2l==1.0.0a1.post0) (0.0.8)
Requirement already satisfied: qtconsole in /usr/local/lib/python3.9/dist-
packages (from jupyter->d2l==1.0.0a1.post0) (5.4.2)
Requirement already satisfied: ipywidgets in /usr/local/lib/python3.9/dist-
packages (from jupyter->d2l==1.0.0a1.post0) (7.7.1)
Requirement already satisfied: jupyter-console in /usr/local/lib/python3.9/dist-
packages (from jupyter->d2l==1.0.0a1.post0) (6.1.0)
Requirement already satisfied: ipykernel in /usr/local/lib/python3.9/dist-
packages (from jupyter->d2l==1.0.0a1.post0) (5.5.6)
Requirement already satisfied: notebook in /usr/local/lib/python3.9/dist-
packages (from jupyter->d2l==1.0.0a1.post0) (6.4.8)
Requirement already satisfied: nbconvert in /usr/local/lib/python3.9/dist-
packages (from jupyter->d2l==1.0.0a1.post0) (6.5.4)
Requirement already satisfied: pyparsing>=2.3.1 in
```

```
/usr/local/lib/python3.9/dist-packages (from matplotlib->d2l==1.0.0a1.post0)
(3.0.9)
Requirement already satisfied: fonttools>=4.22.0 in
/usr/local/lib/python3.9/dist-packages (from matplotlib->d2l==1.0.0a1.post0)
(4.39.3)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.9/dist-
packages (from matplotlib->d2l==1.0.0a1.post0) (23.0)
Requirement already satisfied: importlib-resources>=3.2.0 in
/usr/local/lib/python3.9/dist-packages (from matplotlib->d2l==1.0.0a1.post0)
(5.12.0)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.9/dist-
packages (from matplotlib->d2l==1.0.0a1.post0) (0.11.0)
Requirement already satisfied: python-dateutil>=2.7 in
/usr/local/lib/python3.9/dist-packages (from matplotlib->d2l==1.0.0a1.post0)
(2.8.2)
Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.9/dist-
packages (from matplotlib->d2l==1.0.0a1.post0) (8.4.0)
Requirement already satisfied: contourpy>=1.0.1 in
/usr/local/lib/python3.9/dist-packages (from matplotlib->d2l==1.0.0a1.post0)
(1.0.7)
Requirement already satisfied: kiwisolver>=1.0.1 in
/usr/local/lib/python3.9/dist-packages (from matplotlib->d2l==1.0.0a1.post0)
(1.4.4)
Requirement already satisfied: traitlets in /usr/local/lib/python3.9/dist-
packages (from matplotlib-inline->d2l==1.0.0a1.post0) (5.7.1)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.9/dist-
packages (from pandas->d2l==1.0.0a1.post0) (2022.7.1)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in
/usr/local/lib/python3.9/dist-packages (from requests->d2l==1.0.0a1.post0)
(1.26.15)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.9/dist-
packages (from requests->d2l==1.0.0a1.post0) (3.4)
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.9/dist-packages (from requests->d2l==1.0.0a1.post0)
(2022.12.7)
Requirement already satisfied: charset-normalizer~=2.0.0 in
/usr/local/lib/python3.9/dist-packages (from requests->d2l==1.0.0a1.post0)
(2.0.12)
Requirement already satisfied: zipp>=0.5 in /usr/local/lib/python3.9/dist-
packages (from importlib-metadata>=4.8.0->gym->d2l==1.0.0a1.post0) (3.15.0)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.9/dist-
packages (from python-dateutil>=2.7->matplotlib->d2l==1.0.0a1.post0) (1.16.0)
Requirement already satisfied: jupyter-client in /usr/local/lib/python3.9/dist-
packages (from ipykernel->jupyter->d2l==1.0.0a1.post0) (6.1.12)
Requirement already satisfied: tornado>=4.2 in /usr/local/lib/python3.9/dist-
packages (from ipykernel->jupyter->d2l==1.0.0a1.post0) (6.2)
Requirement already satisfied: ipython-genutils in
/usr/local/lib/python3.9/dist-packages (from
```

```
ipykernel->jupyter->d2l==1.0.0a1.post0) (0.2.0)
Requirement already satisfied: ipython>=5.0.0 in /usr/local/lib/python3.9/dist-
packages (from ipykernel->jupyter->d2l==1.0.0a1.post0) (7.34.0)
Requirement already satisfied: jupyterlab-widgets>=1.0.0 in
/usr/local/lib/python3.9/dist-packages (from
ipywidgets->jupyter->d2l==1.0.0a1.post0) (3.0.7)
Requirement already satisfied: widgetsnbextension~=3.6.0 in
/usr/local/lib/python3.9/dist-packages (from
ipywidgets->jupyter->d2l==1.0.0a1.post0) (3.6.4)
Requirement already satisfied: pygments in /usr/local/lib/python3.9/dist-
packages (from jupyter-console->jupyter->d2l==1.0.0a1.post0) (2.14.0)
Requirement already satisfied: prompt-toolkit!=3.0.0,!=3.0.1,<3.1.0,>=2.0.0 in
/usr/local/lib/python3.9/dist-packages (from jupyter-
console->jupyter->d2l==1.0.0a1.post0) (3.0.38)
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.9/dist-
packages (from nbconvert->jupyter->d2l==1.0.0a1.post0) (2.1.2)
Requirement already satisfied: bleach in /usr/local/lib/python3.9/dist-packages
(from nbconvert->jupyter->d2l==1.0.0a1.post0) (6.0.0)
Requirement already satisfied: jinja2>=3.0 in /usr/local/lib/python3.9/dist-
packages (from nbconvert->jupyter->d2l==1.0.0a1.post0) (3.1.2)
Requirement already satisfied: pandocfilters>=1.4.1 in
/usr/local/lib/python3.9/dist-packages (from
nbconvert->jupyter->d2l==1.0.0a1.post0) (1.5.0)
Requirement already satisfied: entrypoints>=0.2.2 in
/usr/local/lib/python3.9/dist-packages (from
nbconvert->jupyter->d2l==1.0.0a1.post0) (0.4)
Requirement already satisfied: mistune<2,>=0.8.1 in
/usr/local/lib/python3.9/dist-packages (from
nbconvert->jupyter->d2l==1.0.0a1.post0) (0.8.4)
Requirement already satisfied: jupyterlab-pygments in
/usr/local/lib/python3.9/dist-packages (from
nbconvert->jupyter->d2l==1.0.0a1.post0) (0.2.2)
Requirement already satisfied: jupyter-core>=4.7 in
/usr/local/lib/python3.9/dist-packages (from
nbconvert->jupyter->d2l==1.0.0a1.post0) (5.3.0)
Requirement already satisfied: nbformat>=5.1 in /usr/local/lib/python3.9/dist-
packages (from nbconvert->jupyter->d2l==1.0.0a1.post0) (5.8.0)
Requirement already satisfied: nbclient>=0.5.0 in /usr/local/lib/python3.9/dist-
packages (from nbconvert->jupyter->d2l==1.0.0a1.post0) (0.7.3)
Requirement already satisfied: lxml in /usr/local/lib/python3.9/dist-packages
(from nbconvert->jupyter->d2l==1.0.0a1.post0) (4.9.2)
Requirement already satisfied: beautifulsoup4 in /usr/local/lib/python3.9/dist-
packages (from nbconvert->jupyter->d2l==1.0.0a1.post0) (4.11.2)
Requirement already satisfied: tinycss2 in /usr/local/lib/python3.9/dist-
packages (from nbconvert->jupyter->d2l==1.0.0a1.post0) (1.2.1)
Requirement already satisfied: defusedxml in /usr/local/lib/python3.9/dist-
packages (from nbconvert->jupyter->d2l==1.0.0a1.post0) (0.7.1)
Requirement already satisfied: argon2-cffi in /usr/local/lib/python3.9/dist-
```

```
packages (from notebook->jupyter->d2l==1.0.0a1.post0) (21.3.0)
Requirement already satisfied: Send2Trash>=1.8.0 in
/usr/local/lib/python3.9/dist-packages (from
notebook->jupyter->d2l==1.0.0a1.post0) (1.8.0)
Requirement already satisfied: prometheus-client in
/usr/local/lib/python3.9/dist-packages (from
notebook->jupyter->d2l==1.0.0a1.post0) (0.16.0)
Requirement already satisfied: pyzmq>=17 in /usr/local/lib/python3.9/dist-
packages (from notebook->jupyter->d2l==1.0.0a1.post0) (23.2.1)
Requirement already satisfied: nest-asyncio>=1.5 in
/usr/local/lib/python3.9/dist-packages (from
notebook->jupyter->d2l==1.0.0a1.post0) (1.5.6)
Requirement already satisfied: terminado>=0.8.3 in
/usr/local/lib/python3.9/dist-packages (from
notebook->jupyter->d2l==1.0.0a1.post0) (0.17.1)
Requirement already satisfied: qtpy>=2.0.1 in /usr/local/lib/python3.9/dist-
packages (from qtconsole->jupyter->d2l==1.0.0a1.post0) (2.3.1)
Requirement already satisfied: setuptools>=18.5 in
/usr/local/lib/python3.9/dist-packages (from
ipython>=5.0.0->ipykernel->jupyter->d2l==1.0.0a1.post0) (67.6.1)
Requirement already satisfied: pickleshare in /usr/local/lib/python3.9/dist-
packages (from ipython>=5.0.0->ipykernel->jupyter->d2l==1.0.0a1.post0) (0.7.5)
Requirement already satisfied: pexpect>4.3 in /usr/local/lib/python3.9/dist-
packages (from ipython>=5.0.0->ipykernel->jupyter->d2l==1.0.0a1.post0) (4.8.0)
Requirement already satisfied: decorator in /usr/local/lib/python3.9/dist-
packages (from ipython>=5.0.0->ipykernel->jupyter->d2l==1.0.0a1.post0) (4.4.2)
Requirement already satisfied: jedi>=0.16 in /usr/local/lib/python3.9/dist-
packages (from ipython>=5.0.0->ipykernel->jupyter->d2l==1.0.0a1.post0) (0.18.2)
Requirement already satisfied: backcall in /usr/local/lib/python3.9/dist-
packages (from ipython>=5.0.0->ipykernel->jupyter->d2l==1.0.0a1.post0) (0.2.0)
Requirement already satisfied: platformdirs>=2.5 in
/usr/local/lib/python3.9/dist-packages (from jupyter-
core>=4.7->nbconvert->jupyter->d2l==1.0.0a1.post0) (3.2.0)
Requirement already satisfied: fastjsonschema in /usr/local/lib/python3.9/dist-
packages (from nbformat>=5.1->nbconvert->jupyter->d2l==1.0.0a1.post0) (2.16.3)
Requirement already satisfied: jsonschema>=2.6 in /usr/local/lib/python3.9/dist-
packages (from nbformat>=5.1->nbconvert->jupyter->d2l==1.0.0a1.post0) (4.3.3)
Requirement already satisfied: wcwidth in /usr/local/lib/python3.9/dist-packages
(from prompt-toolkit!=3.0.0,!=3.0.1,<3.1.0,>=2.0.0->jupyter-
console->jupyter->d2l==1.0.0a1.post0) (0.2.6)
Requirement already satisfied: ptyprocess in /usr/local/lib/python3.9/dist-
packages (from terminado>=0.8.3->notebook->jupyter->d2l==1.0.0a1.post0) (0.7.0)
Requirement already satisfied: argon2-cffi-bindings in
/usr/local/lib/python3.9/dist-packages (from
argon2-cffi->notebook->jupyter->d2l==1.0.0a1.post0) (21.2.0)
Requirement already satisfied: soupsieve>1.2 in /usr/local/lib/python3.9/dist-
packages (from beautifulsoup4->nbconvert->jupyter->d2l==1.0.0a1.post0) (2.4)
Requirement already satisfied: webencodings in /usr/local/lib/python3.9/dist-
```

```
packages (from bleach->nbconvert->jupyter->d2l==1.0.0a1.post0) (0.5.1)
Requirement already satisfied: parso<0.9.0,>=0.8.0 in
/usr/local/lib/python3.9/dist-packages (from
jedi>=0.16->ipython>=5.0.0->ipykernel->jupyter->d2l==1.0.0a1.post0) (0.8.3)
Requirement already satisfied: pyrsistent!=0.17.0,!=0.17.1,!=0.17.2,>=0.14.0 in
/usr/local/lib/python3.9/dist-packages (from
jsonschema>=2.6->nbformat>=5.1->nbconvert->jupyter->d2l==1.0.0a1.post0) (0.19.3)
Requirement already satisfied: attrs>=17.4.0 in /usr/local/lib/python3.9/dist-
packages (from
jsonschema>=2.6->nbformat>=5.1->nbconvert->jupyter->d2l==1.0.0a1.post0) (22.2.0)
Requirement already satisfied: cffi>=1.0.1 in /usr/local/lib/python3.9/dist-
packages (from argon2-cffi-
bindings->argon2-cffi->notebook->jupyter->d2l==1.0.0a1.post0) (1.15.1)
Requirement already satisfied: pycparser in /usr/local/lib/python3.9/dist-
packages (from cffi>=1.0.1->argon2-cffi-
bindings->argon2-cffi->notebook->jupyter->d2l==1.0.0a1.post0) (2.21)
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-
wheels/public/simple/
Requirement already satisfied: matplotlib_inline in
/usr/local/lib/python3.9/dist-packages (0.1.6)
Requirement already satisfied: traitlets in /usr/local/lib/python3.9/dist-
packages (from matplotlib_inline) (5.7.1)
```

[45]:
```
import collections
import math
import torch
from torch import nn
from torch.nn import functional as F
from d2l import torch as d2l

d2l.use_svg_display()
```

The data:

[46]:
```
class MTFraEng(d2l.DataModule):
    """The English-French dataset."""
    def _download(self):
        d2l.extract(d2l.download(
            d2l.DATA_URL+'fra-eng.zip', self.root,
            '94646ad1522d915e7b0f9296181140edcf86a4f5'))
        with open(self.root + '/fra-eng/fra.txt', encoding='utf-8') as f:
            return f.read()

data = MTFraEng()
raw_text = data._download()
print(raw_text[:75])
```

```
Go.     Va !
```

```
Hi.	Salut !
Run!	Cours !
Run!	Courez !
Who?	Qui ?
Wow!	Ça alors !
```

[47]:
```
@d2l.add_to_class(MTFraEng)
def _preprocess(self, text):
    # Replace non-breaking space with space
    text = text.replace('\u202f', ' ').replace('\xa0', ' ')
    # Insert space between words and punctuation marks
    no_space = lambda char, prev_char: char in ',.!?' and prev_char != ' '
    out = [' ' + char if i > 0 and no_space(char, text[i - 1]) else char
           for i, char in enumerate(text.lower())]
    return ''.join(out)

text = data._preprocess(raw_text)
print(text[:80])
```

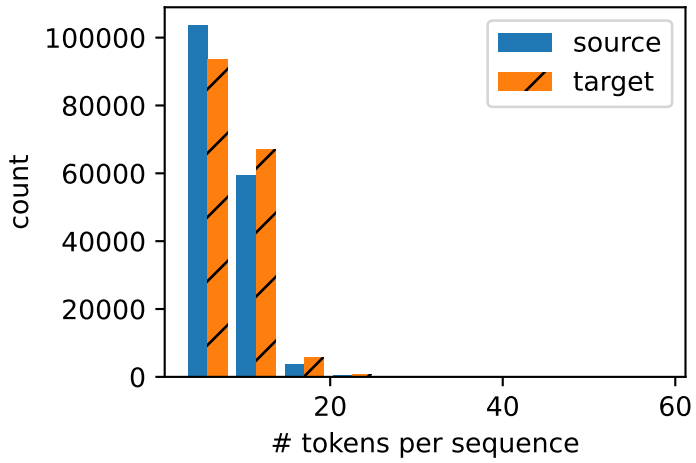
```
go .	va !
hi .	salut !
run !	cours !
run !	courez !
who ?	qui ?
wow !	ça alors !
```

[48]:
```
@d2l.add_to_class(MTFraEng)
def _tokenize(self, text, max_examples=None):
    src, tgt = [], []
    for i, line in enumerate(text.split('\n')):
        if max_examples and i > max_examples: break
        parts = line.split('\t')
        if len(parts) == 2:
            # Skip empty tokens
            src.append([t for t in f'{parts[0]} <eos>'.split(' ') if t])
            tgt.append([t for t in f'{parts[1]} <eos>'.split(' ') if t])
    return src, tgt

src, tgt = data._tokenize(text)
src[:6], tgt[:6]
```

[48]:
```
([['go', '.', '<eos>'],
  ['hi', '.', '<eos>'],
  ['run', '!', '<eos>'],
  ['run', '!', '<eos>'],
  ['who', '?', '<eos>'],
```

```
 ['wow', '!', '<eos>']],
[['va', '!', '<eos>'],
 ['salut', '!', '<eos>'],
 ['cours', '!', '<eos>'],
 ['courez', '!', '<eos>'],
 ['qui', '?', '<eos>'],
 ['ça', 'alors', '!', '<eos>']])
```

[49]:
```
def show_list_len_pair_hist(legend, xlabel, ylabel, xlist, ylist):
    """Plot the histogram for list length pairs."""
    d2l.set_figsize()
    _, _, patches = d2l.plt.hist(
        [[len(l) for l in xlist], [len(l) for l in ylist]])
    d2l.plt.xlabel(xlabel)
    d2l.plt.ylabel(ylabel)
    for patch in patches[1].patches:
        patch.set_hatch('/')
    d2l.plt.legend(legend)

show_list_len_pair_hist(['source', 'target'], '# tokens per sequence',
                        'count', src, tgt);
```



[50]:
```
@d2l.add_to_class(MTFraEng)
def __init__(self, batch_size, num_steps=9, num_train=512, num_val=128):
    super(MTFraEng, self).__init__()
    self.save_hyperparameters()
    self.arrays, self.src_vocab, self.tgt_vocab = self._build_arrays(
        self._download())

@d2l.add_to_class(MTFraEng)
```

```
def _build_arrays(self, raw_text, src_vocab=None, tgt_vocab=None):
    def _build_array(sentences, vocab, is_tgt=False):
        pad_or_trim = lambda seq, t: (
            seq[:t] if len(seq) > t else seq + ['<pad>'] * (t - len(seq)))
        sentences = [pad_or_trim(s, self.num_steps) for s in sentences]
        if is_tgt:
            sentences = [['<bos>'] + s for s in sentences]
        if vocab is None:
            vocab = d2l.Vocab(sentences, min_freq=2)
        array = torch.tensor([vocab[s] for s in sentences])
        valid_len = (array != vocab['<pad>']).type(torch.int32).sum(1)
        return array, vocab, valid_len
    src, tgt = self._tokenize(self._preprocess(raw_text),
                              self.num_train + self.num_val)
    src_array, src_vocab, src_valid_len = _build_array(src, src_vocab)
    tgt_array, tgt_vocab, _ = _build_array(tgt, tgt_vocab, True)
    return ((src_array, tgt_array[:,:-1], src_valid_len, tgt_array[:,1:]),
            src_vocab, tgt_vocab)
```

[51]:
```
@d2l.add_to_class(MTFraEng)
def __init__(self, batch_size, num_steps=9, num_train=512, num_val=128):
    super(MTFraEng, self).__init__()
    self.save_hyperparameters()
    self.arrays, self.src_vocab, self.tgt_vocab = self._build_arrays(
        self._download())

@d2l.add_to_class(MTFraEng)
def _build_arrays(self, raw_text, src_vocab=None, tgt_vocab=None):
    def _build_array(sentences, vocab, is_tgt=False):
        pad_or_trim = lambda seq, t: (
            seq[:t] if len(seq) > t else seq + ['<pad>'] * (t - len(seq)))
        sentences = [pad_or_trim(s, self.num_steps) for s in sentences]
        if is_tgt:
            sentences = [['<bos>'] + s for s in sentences]
        if vocab is None:
            vocab = d2l.Vocab(sentences, min_freq=2)
        array = torch.tensor([vocab[s] for s in sentences])
        valid_len = (array != vocab['<pad>']).type(torch.int32).sum(1)
        return array, vocab, valid_len
    src, tgt = self._tokenize(self._preprocess(raw_text),
                              self.num_train + self.num_val)
    src_array, src_vocab, src_valid_len = _build_array(src, src_vocab)
    tgt_array, tgt_vocab, _ = _build_array(tgt, tgt_vocab, True)
    return ((src_array, tgt_array[:,:-1], src_valid_len, tgt_array[:,1:]),
            src_vocab, tgt_vocab)
```

[52]:
```
@d2l.add_to_class(MTFraEng)
def get_dataloader(self, train):
    idx = slice(0, self.num_train) if train else slice(self.num_train, None)
    return self.get_tensorloader(self.arrays, train, idx)
```

[53]:
```
data = MTFraEng(batch_size=3)
src, tgt, src_valid_len, label = next(iter(data.train_dataloader()))
print('source:', src.type(torch.int32))
print('decoder input:', tgt.type(torch.int32))
print('source len excluding pad:', src_valid_len.type(torch.int32))
print('label:', label.type(torch.int32))
```

```
source: tensor([[ 40,  91,   0,   3,   4,   4,   4,   4,   4],
        [ 85,   5,   2,   3,   4,   4,   4,   4,   4],
        [169,  99,   2,   3,   4,   4,   4,   4,   4]], dtype=torch.int32)
decoder input: tensor([[  3,   6, 193,   0,   4,   5,   5,   5,   5],
        [  3, 108,   6,   2,   4,   5,   5,   5,   5],
        [  3,   6, 187,   2,   4,   5,   5,   5,   5]], dtype=torch.int32)
source len excluding pad: tensor([4, 4, 4], dtype=torch.int32)
label: tensor([[  6, 193,   0,   4,   5,   5,   5,   5,   5],
        [108,   6,   2,   4,   5,   5,   5,   5,   5],
        [  6, 187,   2,   4,   5,   5,   5,   5,   5]], dtype=torch.int32)
```

[54]:
```
@d2l.add_to_class(MTFraEng)
def build(self, src_sentences, tgt_sentences):
    raw_text = '\n'.join([src + '\t' + tgt for src, tgt in zip(
        src_sentences, tgt_sentences)])
    arrays, _, _ = self._build_arrays(
        raw_text, self.src_vocab, self.tgt_vocab)
    return arrays

src, tgt, _,  _ = data.build(['hi .'], ['salut .'])
print('source:', data.src_vocab.to_tokens(src[0].type(torch.int32)))
print('target:', data.tgt_vocab.to_tokens(tgt[0].type(torch.int32)))
```

```
source: ['hi', '.', '<eos>', '<pad>', '<pad>', '<pad>', '<pad>', '<pad>',
'<pad>']
target: ['<bos>', 'salut', '.', '<eos>', '<pad>', '<pad>', '<pad>', '<pad>',
'<pad>']
```

1.1 Problem 1

For the problem of Machine Translation using sequence-to-sequence model

1. Can you adjust the hyperparameters to improve the translation results?
2. If the encoder and the decoder differ in the number of layers or the number of hidden units, how can we initialize the hidden state of the decoder? Please run an experiment for it with 3

layers for encoder and 2 layers for decoder. Plot your training results and compare it against the baseline examples from the lectures.

3. Rerun the baseline experiment by replacing GRU with LSTM. Plot your results and compare GRU against LSTM.

The sequence-to-sequence model base implementation:

[55]:
```
import collections
import math
import torch
from torch import nn
from torch.nn import functional as F
from d2l import torch as d2l

d2l.use_svg_display()
```

[56]:
```
def init_seq2seq(module):
    """Initialize weights for Seq2Seq."""
    if type(module) == nn.Linear:
         nn.init.xavier_uniform_(module.weight)
    if type(module) == nn.GRU:
        for param in module._flat_weights_names:
            if "weight" in param:
                nn.init.xavier_uniform_(module._parameters[param])

class Seq2SeqEncoder(d2l.Encoder):
    def __init__(self, vocab_size, embed_size, num_hiddens, num_layers,
                 dropout=0):
        super().__init__()
        self.embedding = nn.Embedding(vocab_size, embed_size)
        self.rnn = d2l.GRU(embed_size, num_hiddens, num_layers, dropout)
        self.apply(init_seq2seq)

    def forward(self, X, *args):
        embs = self.embedding(X.t().type(torch.int64))
        outputs, state = self.rnn(embs)
        return outputs, state

class Seq2SeqDecoder(d2l.Decoder):
    """The RNN decoder for sequence to sequence learning."""
    def __init__(self, vocab_size, embed_size, num_hiddens, num_layers,
                 dropout=0):
        super().__init__()
        self.embedding = nn.Embedding(vocab_size, embed_size)
        self.rnn = d2l.GRU(embed_size+num_hiddens, num_hiddens, num_layers,␣
 ↪dropout)
        self.dense = nn.LazyLinear(vocab_size)
```

```
        self.apply(init_seq2seq)

    def init_state(self, enc_all_outputs, *args):
        return enc_all_outputs

    def forward(self, X, state):
        embs = self.embedding(X.t().type(torch.int32))
        enc_output, hidden_state = state
        context = enc_output[-1]
        context = context.repeat(embs.shape[0], 1, 1)
        embs_and_context = torch.cat((embs, context), -1)
        outputs, hidden_state = self.rnn(embs_and_context, hidden_state)
        outputs = self.dense(outputs).swapaxes(0, 1)
        return outputs, [enc_output, hidden_state]

class Seq2Seq(d2l.EncoderDecoder):
    def __init__(self, encoder, decoder, tgt_pad, lr):
        super().__init__(encoder, decoder)
        self.save_hyperparameters()

    def validation_step(self, batch):
        Y_hat = self(*batch[:-1])
        self.plot('loss', self.loss(Y_hat, batch[-1]), train=False)

    def configure_optimizers(self):
        return torch.optim.Adam(self.parameters(), lr=self.lr)

    def loss(self, Y_hat, Y):
        l = super(Seq2Seq, self).loss(Y_hat, Y, averaged=False)
        mask = (Y.reshape(-1) != self.tgt_pad).type(torch.float32)
        return (l * mask).sum() / mask.sum()

@d2l.add_to_class(d2l.EncoderDecoder)
def predict_step(self, batch, device, num_steps,
                 save_attention_weights=False):
    batch = [a.to(device) for a in batch]
    src, tgt, src_valid_len, _ = batch
    enc_all_outputs = self.encoder(src, src_valid_len)
    dec_state = self.decoder.init_state(enc_all_outputs, src_valid_len)
    outputs, attention_weights = [tgt[:, 0].unsqueeze(1), ], []
    for _ in range(num_steps):
        Y, dec_state = self.decoder(outputs[-1], dec_state)
        outputs.append(Y.argmax(2))
        # Save attention weights (to be covered later)
        if save_attention_weights:
            attention_weights.append(self.decoder.attention_weights)
    return torch.cat(outputs[1:], 1), attention_weights
```

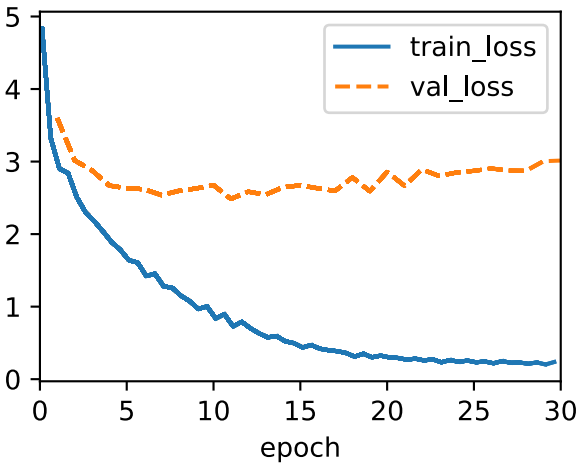
```
def bleu(pred_seq, label_seq, k):
    """Compute the BLEU."""
    pred_tokens, label_tokens = pred_seq.split(' '), label_seq.split(' ')
    len_pred, len_label = len(pred_tokens), len(label_tokens)
    score = math.exp(min(0, 1 - len_label / len_pred))
    for n in range(1, min(k, len_pred) + 1):
        num_matches, label_subs = 0, collections.defaultdict(int)
        for i in range(len_label - n + 1):
            label_subs[' '.join(label_tokens[i: i + n])] += 1
        for i in range(len_pred - n + 1):
            if label_subs[' '.join(pred_tokens[i: i + n])] > 0:
                num_matches += 1
                label_subs[' '.join(pred_tokens[i: i + n])] -= 1
        score *= math.pow(num_matches / (len_pred - n + 1), math.pow(0.5, n))
    return score


vocab_size, embed_size, num_hiddens, num_layers = 10, 8, 16, 2
batch_size, num_steps = 4, 9
encoder = Seq2SeqEncoder(vocab_size, embed_size, num_hiddens, num_layers)
X = torch.zeros((batch_size, num_steps))
enc_outputs, enc_state = encoder(X)
d2l.check_shape(enc_outputs, (num_steps, batch_size, num_hiddens))

d2l.check_shape(enc_state, (num_layers, batch_size, num_hiddens))

decoder = Seq2SeqDecoder(vocab_size, embed_size, num_hiddens, num_layers)
state = decoder.init_state(encoder(X))
dec_outputs, state = decoder(X, state)
d2l.check_shape(dec_outputs, (batch_size, num_steps, vocab_size))
d2l.check_shape(state[1], (num_layers, batch_size, num_hiddens))
```

```
/usr/local/lib/python3.9/dist-packages/torch/nn/modules/lazy.py:180:
UserWarning: Lazy modules are a new feature under heavy development so changes
to the API or functionality can happen at any moment.
  warnings.warn('Lazy modules are a new feature under heavy development '
```

Baseline model:

[57]:
```
data = d2l.MTFraEng(batch_size=128)
embed_size, num_hiddens, num_layers, dropout = 256, 256, 2, 0.2
encoder = Seq2SeqEncoder(
    len(data.src_vocab), embed_size, num_hiddens, num_layers, dropout)
decoder = Seq2SeqDecoder(
    len(data.tgt_vocab), embed_size, num_hiddens, num_layers, dropout)
model = Seq2Seq(encoder, decoder, tgt_pad=data.tgt_vocab['<pad>'],
                lr=0.005)
```

```
trainer = d2l.Trainer(max_epochs=30, gradient_clip_val=1, num_gpus=1)
trainer.fit(model, data)
```



[58]:
```
engs = ['go .', 'i lost .', 'he\'s calm .', 'i\'m home .']
fras = ['va !', 'j\'ai perdu .', 'il est calme .', 'je suis chez moi .']
preds, _ = model.predict_step(
    data.build(engs, fras), d2l.try_gpu(), data.num_steps)
for en, fr, p in zip(engs, fras, preds):
    translation = []
    for token in data.tgt_vocab.to_tokens(p):
        if token == '<eos>':
            break
        translation.append(token)
    print(f'{en} => {translation}, bleu,'
          f'{bleu(" ".join(translation), fr, k=2):.3f}')
```

```
go . => ['va', '!'], bleu,1.000
i lost . => ["j'ai", 'perdu', '.'], bleu,1.000
he's calm . => ['sois', 'calme', '.'], bleu,0.492
i'm home . => ['je', 'suis', 'chez', 'moi', '.'], bleu,1.000
```

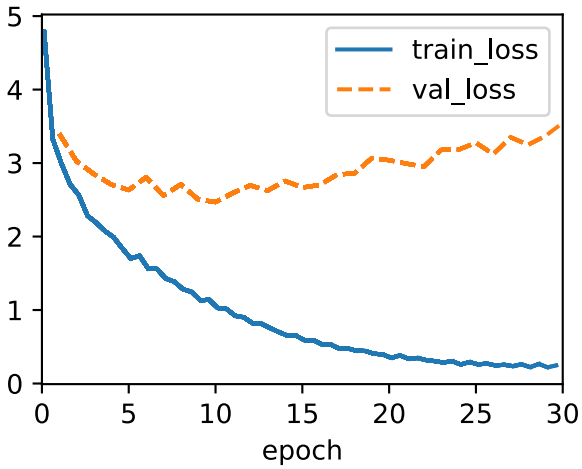
Adjusted:

[59]:
```
class Seq2SeqEncoder2(d2l.Encoder):
    def __init__(self, vocab_size, embed_size, num_hiddens, num_layers,
                 dropout=0):
        super().__init__()
        self.embedding = nn.Embedding(vocab_size, embed_size)
        self.rnn = d2l.GRU(embed_size, num_hiddens, num_layers, dropout)
        self.apply(init_seq2seq)
```

```
    def forward(self, X, *args):
        embs = self.embedding(X.t().type(torch.int64))
        outputs, state = self.rnn(embs)
        return outputs, state

class Seq2SeqDecoder2(d2l.Decoder):
    """The RNN decoder for sequence to sequence learning."""
    def __init__(self, vocab_size, embed_size, num_hiddens, num_layers,␣
 ↪dropout=0):
        super().__init__()
        self.embedding = nn.Embedding(vocab_size, embed_size)
        self.rnn = d2l.GRU(embed_size+num_hiddens, num_hiddens, num_layers,␣
 ↪dropout)
        self.dense = nn.LazyLinear(vocab_size)
        self.apply(init_seq2seq)

    def init_state(self, enc_outputs, *args):
        enc_output, hidden_state = enc_outputs
        hidden_state = hidden_state.mean(dim=0, keepdim=True)
        hidden_state = hidden_state.repeat(self.rnn.num_layers, 1, 1)
        return enc_output, hidden_state

    def forward(self, X, state):
        embs = self.embedding(X.t().type(torch.int32))
        enc_output, hidden_state = state
        context = enc_output[-1]
        context = context.repeat(embs.shape[0], 1, 1)
        embs_and_context = torch.cat((embs, context), -1)
        outputs, hidden_state = self.rnn(embs_and_context, hidden_state)
        outputs = self.dense(outputs).swapaxes(0, 1)
        return outputs, [enc_output, hidden_state]

class Seq2Seq2(d2l.EncoderDecoder):
    def __init__(self, encoder, decoder, tgt_pad, lr):
        super().__init__(encoder, decoder)
        self.save_hyperparameters()

    def validation_step(self, batch):
        Y_hat = self(*batch[:-1])
        self.plot('loss', self.loss(Y_hat, batch[-1]), train=False)

    def configure_optimizers(self):
        return torch.optim.Adam(self.parameters(), lr=self.lr)

    def loss(self, Y_hat, Y):
        l = super(Seq2Seq2, self).loss(Y_hat, Y, averaged=False)
        mask = (Y.reshape(-1) != self.tgt_pad).type(torch.float32)
```

```
        return (l * mask).sum() / mask.sum()


vocab_size, embed_size, num_hiddens, num_layers = 10, 8, 16, 2
batch_size, num_steps = 4, 9
encoder = Seq2SeqEncoder2(vocab_size, embed_size, num_hiddens, num_layers)
X = torch.zeros((batch_size, num_steps))
enc_outputs, enc_state = encoder(X)
d2l.check_shape(enc_outputs, (num_steps, batch_size, num_hiddens))

d2l.check_shape(enc_state, (num_layers, batch_size, num_hiddens))

decoder = Seq2SeqDecoder2(vocab_size, embed_size, num_hiddens, num_layers)
state = decoder.init_state(encoder(X))
dec_outputs, state = decoder(X, state)
d2l.check_shape(dec_outputs, (batch_size, num_steps, vocab_size))
d2l.check_shape(state[1], (num_layers, batch_size, num_hiddens))
```

[60]:
```
data = d2l.MTFraEng(batch_size=128)
embed_size, num_hiddens, num_layers, dropout = 256, 256, 2, 0.2

encoder = Seq2SeqEncoder2(len(data.src_vocab), embed_size, num_hiddens,␣
 ↪num_layers=3, dropout=dropout)
decoder = Seq2SeqDecoder2(len(data.tgt_vocab), embed_size, num_hiddens,␣
 ↪num_layers=2, dropout=dropout)
model = Seq2Seq2(encoder, decoder, tgt_pad=data.tgt_vocab['<pad>'], lr=0.005)

trainer = d2l.Trainer(max_epochs=30, gradient_clip_val=1, num_gpus=1)
trainer.fit(model, data)
```



[61]:
```
engs = ['go .', 'i lost .', 'he\'s calm .', 'i\'m home .']
fras = ['va !', 'j\'ai perdu .', 'il est calme .', 'je suis chez moi .']
preds, _ = model.predict_step(
    data.build(engs, fras), d2l.try_gpu(), data.num_steps)
for en, fr, p in zip(engs, fras, preds):
    translation = []
    for token in data.tgt_vocab.to_tokens(p):
        if token == '<eos>':
            break
        translation.append(token)
    print(f'{en} => {translation}, bleu,'
          f'{bleu(" ".join(translation), fr, k=2):.3f}')
```

```
go . => ['va', '!'], bleu,1.000
i lost . => ["j'ai", 'perdu', '.'], bleu,1.000
he's calm . => ['il', 'est', 'mouillé', '.'], bleu,0.658
i'm home . => ['je', 'suis', 'chez', 'moi', '.'], bleu,1.000
```

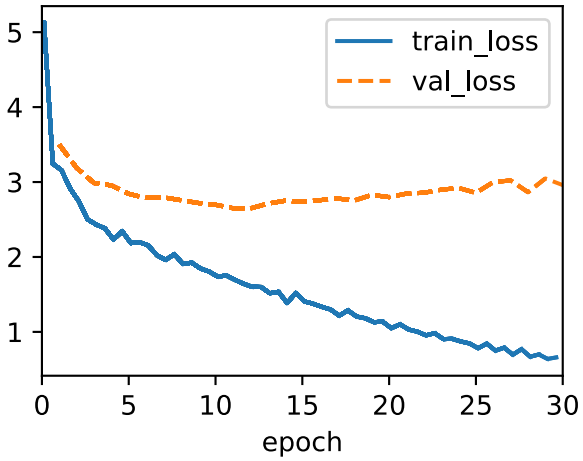
Adjusted with LSTM:

[62]:
```
class LSTM(d2l.RNN):
    def __init__(self, num_inputs, num_hiddens, num_layers, dropout=0):
        d2l.Module.__init__(self)
        self.save_hyperparameters()
        self.rnn = nn.LSTM(num_inputs, num_hiddens, num_layers=num_layers,␣
 ↪dropout=dropout)

    def forward(self, inputs, H_C=None):
        return self.rnn(inputs, H_C)

class Seq2SeqEncoder3(d2l.Encoder):
    def __init__(self, vocab_size, embed_size, num_hiddens, num_layers,
                 dropout=0):
        super().__init__()
        self.embedding = nn.Embedding(vocab_size, embed_size)
        self.rnn = LSTM(embed_size, num_hiddens, num_layers, dropout)
        self.apply(init_seq2seq)

    def forward(self, X, *args):
        embs = self.embedding(X.t().type(torch.int64))
        outputs, state = self.rnn(embs)
        return outputs, state

class Seq2SeqDecoder3(d2l.Decoder):
    def __init__(self, vocab_size, embed_size, num_hiddens, num_layers,␣
 ↪dropout=0):
        super().__init__()
        self.embedding = nn.Embedding(vocab_size, embed_size)
```

```
        self.rnn = LSTM(embed_size+num_hiddens, num_hiddens, num_layers,␣
 ↪dropout)
        self.dense = nn.LazyLinear(vocab_size)
        self.apply(init_seq2seq)

    def init_state(self, enc_all_outputs, *args):
        return enc_all_outputs

    def forward(self, X, state):
        embs = self.embedding(X.t().type(torch.int32))
        enc_output, hidden_state = state
        context = enc_output[-1]
        context = context.repeat(embs.shape[0], 1, 1)
        embs_and_context = torch.cat((embs, context), -1)
        outputs, hidden_state = self.rnn(embs_and_context, hidden_state)
        outputs = self.dense(outputs).swapaxes(0, 1)
        return outputs, [enc_output, hidden_state]

class Seq2Seq3(d2l.EncoderDecoder):
    def __init__(self, encoder, decoder, tgt_pad, lr):
        super().__init__(encoder, decoder)
        self.save_hyperparameters()

    def validation_step(self, batch):
        Y_hat = self(*batch[:-1])
        self.plot('loss', self.loss(Y_hat, batch[-1]), train=False)

    def configure_optimizers(self):
        return torch.optim.Adam(self.parameters(), lr=self.lr)

    def loss(self, Y_hat, Y):
        l = super(Seq2Seq3, self).loss(Y_hat, Y, averaged=False)
        mask = (Y.reshape(-1) != self.tgt_pad).type(torch.float32)
        return (l * mask).sum() / mask.sum()


vocab_size, embed_size, num_hiddens, num_layers = 10, 8, 16, 2
batch_size, num_steps = 4, 9
encoder = Seq2SeqEncoder3(vocab_size, embed_size, num_hiddens, num_layers)
X = torch.zeros((batch_size, num_steps))
enc_outputs, enc_state = encoder(X)
d2l.check_shape(enc_outputs, (num_steps, batch_size, num_hiddens))

#d2l.check_shape(enc_state, (num_layers, batch_size, num_hiddens))

decoder = Seq2SeqDecoder3(vocab_size, embed_size, num_hiddens, num_layers)
state = decoder.init_state(encoder(X))
```

```
dec_outputs, state = decoder(X, state)
d2l.check_shape(dec_outputs, (batch_size, num_steps, vocab_size))
#d2l.check_shape(state[1], (num_layers, batch_size, num_hiddens))
```

[63]:
```
data = d2l.MTFraEng(batch_size=128)
embed_size, num_hiddens, num_layers, dropout = 256, 256, 2, 0.2

encoder = Seq2SeqEncoder3(len(data.src_vocab), embed_size, num_hiddens,␣
 ↪num_layers, dropout=dropout)
decoder = Seq2SeqDecoder3(len(data.tgt_vocab), embed_size, num_hiddens,␣
 ↪num_layers, dropout=dropout)
model = Seq2Seq3(encoder, decoder, tgt_pad=data.tgt_vocab['<pad>'], lr=0.005)

trainer = d2l.Trainer(max_epochs=30, gradient_clip_val=1, num_gpus=1)
trainer.fit(model, data)
```



[64]:
```
engs = ['go .', 'i lost .', 'he\'s calm .', 'i\'m home .']
fras = ['va !', 'j\'ai perdu .', 'il est calme .', 'je suis chez moi .']
preds, _ = model.predict_step(
    data.build(engs, fras), d2l.try_gpu(), data.num_steps)
for en, fr, p in zip(engs, fras, preds):
    translation = []
    for token in data.tgt_vocab.to_tokens(p):
        if token == '<eos>':
            break
        translation.append(token)
    print(f'{en} => {translation}, bleu,'
          f'{bleu(" ".join(translation), fr, k=2):.3f}')
```

```
go . => ['<unk>', '!'], bleu,0.000
i lost . => ['je', 'me', 'suis', '<unk>', '.'], bleu,0.000
```

```
he's calm . => ['sois', 'calme', '!'], bleu,0.000
i'm home . => ['je', 'suis', '<unk>', '.'], bleu,0.512
```

1.2 Problem 2

For the problem of Machine Translation with Bahdanau attention based sequence-to-sequence modeling

1. Explore the impacts of number of hidden layers starting tom 1 hidden layer up to 4 hidden layers. Plot the results (training loss and validation), also run few examples to do the qualitative comparison between these two. Can you draw the attention weight matrixes and compare them.
2. Replace GRU with LSTM in the experiment. Perform training again. Plot the results (training loss and validation), also run few examples to do the qualitative comparison between these two. Can you draw the attention weight matrixes and compare them.

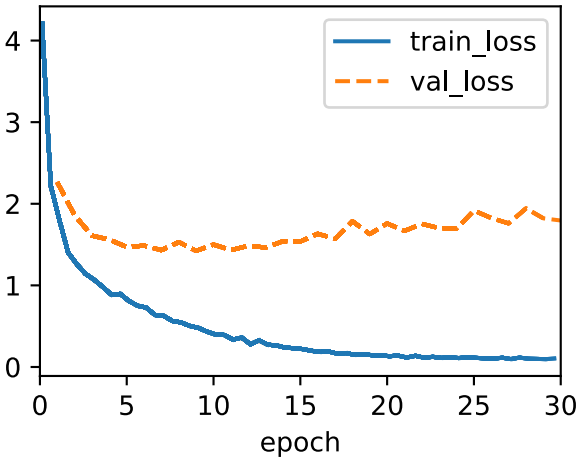
[65]:
```python
class AttentionDecoder(d2l.Decoder):
    """The base attention-based decoder interface."""
    def __init__(self):
        super().__init__()

    @property
    def attention_weights(self):
        raise NotImplementedError

class Seq2SeqAttentionDecoder(AttentionDecoder):
    def __init__(self, vocab_size, embed_size, num_hiddens, num_layers,
                 dropout=0):
        super().__init__()
        self.attention = d2l.AdditiveAttention(num_hiddens, dropout)
        self.embedding = nn.Embedding(vocab_size, embed_size)
        self.rnn = nn.GRU(
            embed_size + num_hiddens, num_hiddens, num_layers,
            dropout=dropout)
        self.dense = nn.LazyLinear(vocab_size)
        self.apply(d2l.init_seq2seq)

    def init_state(self, enc_outputs, enc_valid_lens):
        # Shape of outputs: (num_steps, batch_size, num_hiddens).
        # Shape of hidden_state: (num_layers, batch_size, num_hiddens)
        outputs, hidden_state = enc_outputs
        return (outputs.permute(1, 0, 2), hidden_state, enc_valid_lens)

    def forward(self, X, state):
        # Shape of enc_outputs: (batch_size, num_steps, num_hiddens).
        # Shape of hidden_state: (num_layers, batch_size, num_hiddens)
        enc_outputs, hidden_state, enc_valid_lens = state
        # Shape of the output X: (num_steps, batch_size, embed_size)
```

```
        X = self.embedding(X).permute(1, 0, 2)
        outputs, self._attention_weights = [], []
        for x in X:
            # Shape of query: (batch_size, 1, num_hiddens)
            query = torch.unsqueeze(hidden_state[-1], dim=1)
            # Shape of context: (batch_size, 1, num_hiddens)
            context = self.attention(
                query, enc_outputs, enc_outputs, enc_valid_lens)
            # Concatenate on the feature dimension
            x = torch.cat((context, torch.unsqueeze(x, dim=1)), dim=-1)
            # Reshape x as (1, batch_size, embed_size + num_hiddens)
            out, hidden_state = self.rnn(x.permute(1, 0, 2), hidden_state)
            outputs.append(out)
            self._attention_weights.append(self.attention.attention_weights)
        # After fully connected layer transformation, shape of outputs:
        # (num_steps, batch_size, vocab_size)
        outputs = self.dense(torch.cat(outputs, dim=0))
        return outputs.permute(1, 0, 2), [enc_outputs, hidden_state,
                                          enc_valid_lens]

    @property
    def attention_weights(self):
        return self._attention_weights

vocab_size, embed_size, num_hiddens, num_layers = 10, 8, 16, 2
batch_size, num_steps = 4, 7
encoder = d2l.Seq2SeqEncoder(vocab_size, embed_size, num_hiddens, num_layers)
decoder = Seq2SeqAttentionDecoder(vocab_size, embed_size, num_hiddens,
                                  num_layers)
X = torch.zeros((batch_size, num_steps), dtype=torch.long)
state = decoder.init_state(encoder(X), None)
output, state = decoder(X, state)
d2l.check_shape(output, (batch_size, num_steps, vocab_size))
d2l.check_shape(state[0], (batch_size, num_steps, num_hiddens))
d2l.check_shape(state[1][0], (batch_size, num_hiddens))
```

Base line:

[66]:
```
data = d2l.MTFraEng(batch_size=128)
embed_size, num_hiddens, num_layers, dropout = 256, 256, 2, 0.2

encoder = d2l.Seq2SeqEncoder(len(data.src_vocab), embed_size, num_hiddens,␣
 ↪num_layers, dropout)
decoder = Seq2SeqAttentionDecoder(len(data.tgt_vocab), embed_size, num_hiddens,␣
 ↪num_layers, dropout)
model = d2l.Seq2Seq(encoder, decoder, tgt_pad=data.tgt_vocab['<pad>'], lr=0.005)
```

```
trainer = d2l.Trainer(max_epochs=30, gradient_clip_val=1, num_gpus=1)
trainer.fit(model, data)
```



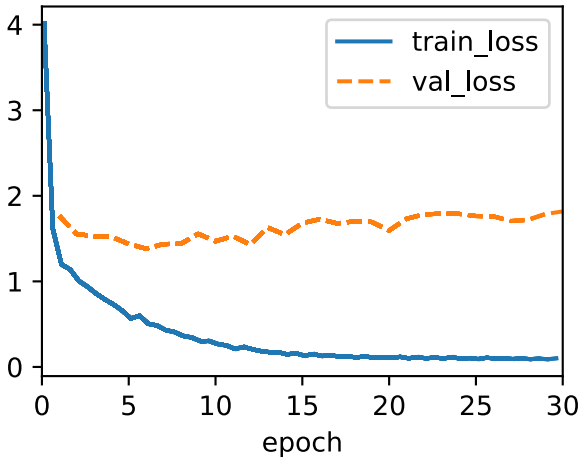
[67]:
```
engs = ['go .', 'i lost .', 'he\'s calm .', 'i\'m home .']
fras = ['va !', 'j\'ai perdu .', 'il est calme .', 'je suis chez moi .']
preds, _ = model.predict_step(
    data.build(engs, fras), d2l.try_gpu(), data.num_steps)
for en, fr, p in zip(engs, fras, preds):
    translation = []
    for token in data.tgt_vocab.to_tokens(p):
        if token == '<eos>':
            break
        translation.append(token)
    print(f'{en} => {translation}, bleu,'
          f'{bleu(" ".join(translation), fr, k=2):.3f}')
```

```
go . => ['va', '!'], bleu,1.000
i lost . => ["j'ai", 'perdu', '.'], bleu,1.000
he's calm . => ['il', 'est', 'mouillé', '.'], bleu,0.658
i'm home . => ['je', 'suis', 'chez', 'moi', '.'], bleu,1.000
```

1 Layer:

[68]:
```
data = d2l.MTFraEng(batch_size=128)
embed_size, num_hiddens, num_layers, dropout = 256, 256, 1, 0.2

encoder = d2l.Seq2SeqEncoder(len(data.src_vocab), embed_size, num_hiddens,␣
 ↪num_layers, dropout)
decoder = Seq2SeqAttentionDecoder(len(data.tgt_vocab), embed_size, num_hiddens,␣
 ↪num_layers, dropout)
model = d2l.Seq2Seq(encoder, decoder, tgt_pad=data.tgt_vocab['<pad>'], lr=0.005)
```

```
trainer = d2l.Trainer(max_epochs=30, gradient_clip_val=1, num_gpus=1)
trainer.fit(model, data)
```



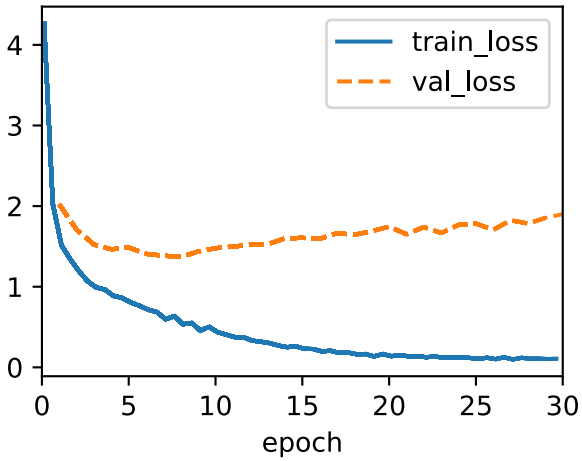
[69]:
```
engs = ['go .', 'i lost .', 'he\'s calm .', 'i\'m home .']
fras = ['va !', 'j\'ai perdu .', 'il est calme .', 'je suis chez moi .']
preds, _ = model.predict_step(
    data.build(engs, fras), d2l.try_gpu(), data.num_steps)
for en, fr, p in zip(engs, fras, preds):
    translation = []
    for token in data.tgt_vocab.to_tokens(p):
        if token == '<eos>':
            break
        translation.append(token)
    print(f'{en} => {translation}, bleu,'
          f'{bleu(" ".join(translation), fr, k=2):.3f}')
```

```
go . => ['va', '!'], bleu,1.000
i lost . => ["j'ai", 'perdu', '.'], bleu,1.000
he's calm . => ['<unk>', '.'], bleu,0.000
i'm home . => ['je', 'suis', 'chez', 'moi', '.'], bleu,1.000
```

2 Layers:

[70]:
```
data = d2l.MTFraEng(batch_size=128)
embed_size, num_hiddens, num_layers, dropout = 256, 256, 2, 0.2

encoder = d2l.Seq2SeqEncoder(len(data.src_vocab), embed_size, num_hiddens,␣
 ↪num_layers, dropout)
decoder = Seq2SeqAttentionDecoder(len(data.tgt_vocab), embed_size, num_hiddens,␣
 ↪num_layers, dropout)
```

```
model = d2l.Seq2Seq(encoder, decoder, tgt_pad=data.tgt_vocab['<pad>'], lr=0.005)

trainer = d2l.Trainer(max_epochs=30, gradient_clip_val=1, num_gpus=1)
trainer.fit(model, data)
```



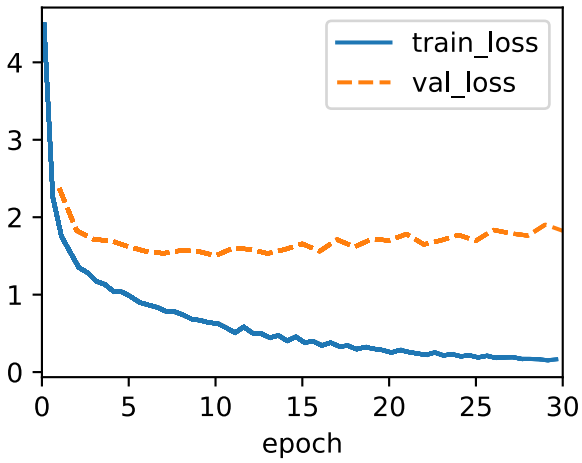
[71]:
```
engs = ['go .', 'i lost .', 'he\'s calm .', 'i\'m home .']
fras = ['va !', 'j\'ai perdu .', 'il est calme .', 'je suis chez moi .']
preds, _ = model.predict_step(
    data.build(engs, fras), d2l.try_gpu(), data.num_steps)
for en, fr, p in zip(engs, fras, preds):
    translation = []
    for token in data.tgt_vocab.to_tokens(p):
        if token == '<eos>':
            break
        translation.append(token)
    print(f'{en} => {translation}, bleu,'
          f'{bleu(" ".join(translation), fr, k=2):.3f}')
```

```
go . => ['va', '!'], bleu,1.000
i lost . => ["j'ai", 'perdu', '.'], bleu,1.000
he's calm . => ['nous', '<unk>', '.'], bleu,0.000
i'm home . => ['je', 'suis', 'chez', 'moi', '.'], bleu,1.000
```

3 Layers:

[72]:
```
data = d2l.MTFraEng(batch_size=128)
embed_size, num_hiddens, num_layers, dropout = 256, 256, 3, 0.2

encoder = d2l.Seq2SeqEncoder(len(data.src_vocab), embed_size, num_hiddens,␣
 ↪num_layers, dropout)
```

```
decoder = Seq2SeqAttentionDecoder(len(data.tgt_vocab), embed_size, num_hiddens,␣
 ↪num_layers, dropout)
model = d2l.Seq2Seq(encoder, decoder, tgt_pad=data.tgt_vocab['<pad>'], lr=0.005)

trainer = d2l.Trainer(max_epochs=30, gradient_clip_val=1, num_gpus=1)
trainer.fit(model, data)
```



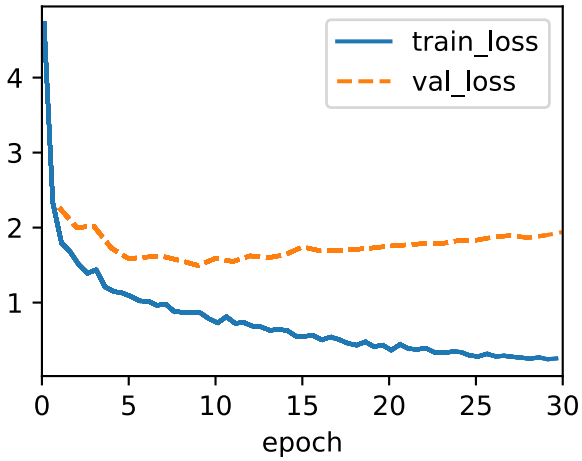
[73]:
```
engs = ['go .', 'i lost .', 'he\'s calm .', 'i\'m home .']
fras = ['va !', 'j\'ai perdu .', 'il est calme .', 'je suis chez moi .']
preds, _ = model.predict_step(
    data.build(engs, fras), d2l.try_gpu(), data.num_steps)
for en, fr, p in zip(engs, fras, preds):
    translation = []
    for token in data.tgt_vocab.to_tokens(p):
        if token == '<eos>':
            break
        translation.append(token)
    print(f'{en} => {translation}, bleu,'
          f'{bleu(" ".join(translation), fr, k=2):.3f}')
```

```
go . => ['va', '!'], bleu,1.000
i lost . => ["j'ai", 'perdu', '.'], bleu,1.000
he's calm . => ['ils', 'ont', 'nous', '.'], bleu,0.000
i'm home . => ['je', 'suis', 'chez', 'moi', '.'], bleu,1.000
```

4 Layers:

[74]:
```
data = d2l.MTFraEng(batch_size=128)
embed_size, num_hiddens, num_layers, dropout = 256, 256, 4, 0.2
```

```
encoder = d2l.Seq2SeqEncoder(len(data.src_vocab), embed_size, num_hiddens,␣
 ↪num_layers, dropout)
decoder = Seq2SeqAttentionDecoder(len(data.tgt_vocab), embed_size, num_hiddens,␣
 ↪num_layers, dropout)
model = d2l.Seq2Seq(encoder, decoder, tgt_pad=data.tgt_vocab['<pad>'], lr=0.005)

trainer = d2l.Trainer(max_epochs=30, gradient_clip_val=1, num_gpus=1)
trainer.fit(model, data)
```



Prediction:

[75]:
```
engs = ['go .', 'i lost .', 'he\'s calm .', 'i\'m home .']
fras = ['va !', 'j\'ai perdu .', 'il est calme .', 'je suis chez moi .']
preds, _ = model.predict_step(
    data.build(engs, fras), d2l.try_gpu(), data.num_steps)
for en, fr, p in zip(engs, fras, preds):
    translation = []
    for token in data.tgt_vocab.to_tokens(p):
        if token == '<eos>':
            break
        translation.append(token)
    print(f'{en} => {translation}, bleu,'
          f'{bleu(" ".join(translation), fr, k=2):.3f}')
```

```
go . => ['vas-y', '!'], bleu,0.000
i lost . => ['je', 'suis', '<unk>', '.'], bleu,0.000
he's calm . => ['il', 'est', '<unk>', '.'], bleu,0.658
i'm home . => ['je', 'suis', 'bien', '.'], bleu,0.512
```

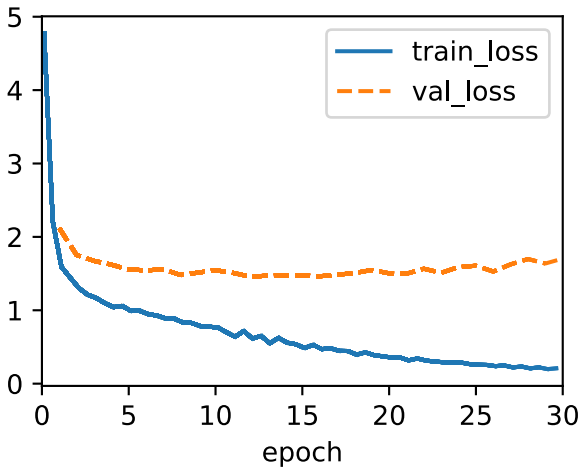
With LSTM:

[76]:
```
# Modified Seq2SeqAttentionDecoder Class for LSTM
class Seq2SeqAttentionDecoder2(AttentionDecoder):
    def __init__(self, vocab_size, embed_size, num_hiddens, num_layers,
                 dropout=0):
        super().__init__()
        self.attention = d2l.AdditiveAttention(num_hiddens, dropout)
        self.embedding = nn.Embedding(vocab_size, embed_size)
        self.rnn = nn.LSTM(  # Change this line
            embed_size + num_hiddens, num_hiddens, num_layers,
            dropout=dropout)
        self.dense = nn.LazyLinear(vocab_size)
        self.apply(d2l.init_seq2seq)

    def init_state(self, enc_outputs, enc_valid_lens):
        outputs, hidden_state = enc_outputs
        # Initialize cell state with the same shape as hidden_state
        cell_state = hidden_state.new_zeros(hidden_state.shape)
        return (outputs.permute(1, 0, 2), (hidden_state, cell_state),␣
↪enc_valid_lens)  # Update this line

    def forward(self, X, state):
        enc_outputs, hidden_and_cell_state, enc_valid_lens = state  # Update␣
↪this line
        X = self.embedding(X).permute(1, 0, 2)
        outputs, self._attention_weights = [], []
        for x in X:
            query = torch.unsqueeze(hidden_and_cell_state[0][-1], dim=1)  #␣
↪Update this line
            context = self.attention(
                query, enc_outputs, enc_outputs, enc_valid_lens)
            x = torch.cat((context, torch.unsqueeze(x, dim=1)), dim=-1)
            out, hidden_and_cell_state = self.rnn(x.permute(1, 0, 2),␣
↪hidden_and_cell_state)  # Update this line
            outputs.append(out)
            self._attention_weights.append(self.attention.attention_weights)
        outputs = self.dense(torch.cat(outputs, dim=0))
        return outputs.permute(1, 0, 2), [enc_outputs, hidden_and_cell_state,
                                          enc_valid_lens]  # Update this line

    @property
    def attention_weights(self):
        return self._attention_weights
```

Baseline:

[77]:
```
data = d2l.MTFraEng(batch_size=128)
embed_size, num_hiddens, num_layers, dropout = 256, 256, 2, 0.2
```

```
encoder = d2l.Seq2SeqEncoder(len(data.src_vocab), embed_size, num_hiddens,␣
 ↪num_layers, dropout)
decoder = Seq2SeqAttentionDecoder2(len(data.tgt_vocab), embed_size,␣
 ↪num_hiddens, num_layers, dropout)
model = d2l.Seq2Seq(encoder, decoder, tgt_pad=data.tgt_vocab['<pad>'], lr=0.005)

trainer = d2l.Trainer(max_epochs=30, gradient_clip_val=1, num_gpus=1)
trainer.fit(model, data)
```



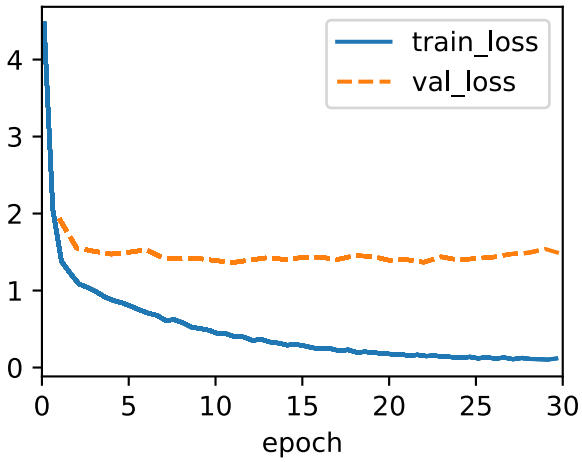
Prediction:

[78]:
```
engs = ['go .', 'i lost .', 'he\'s calm .', 'i\'m home .']
fras = ['va !', 'j\'ai perdu .', 'il est calme .', 'je suis chez moi .']
preds, _ = model.predict_step(
    data.build(engs, fras), d2l.try_gpu(), data.num_steps)
for en, fr, p in zip(engs, fras, preds):
    translation = []
    for token in data.tgt_vocab.to_tokens(p):
        if token == '<eos>':
            break
        translation.append(token)
    print(f'{en} => {translation}, bleu,'
          f'{bleu(" ".join(translation), fr, k=2):.3f}')
```

```
go . => ['va', '!'], bleu,1.000
i lost . => ['je', "l'ai", 'emporté', '.'], bleu,0.000
he's calm . => ['il', 'est', 'mouillé', '.'], bleu,0.658
i'm home . => ['je', 'suis', 'détendu', '.'], bleu,0.512
```

1 Layer:

[79]:
```
data = d2l.MTFraEng(batch_size=128)
embed_size, num_hiddens, num_layers, dropout = 256, 256, 1, 0.2

encoder = d2l.Seq2SeqEncoder(len(data.src_vocab), embed_size, num_hiddens,␣
 ↪num_layers, dropout)
decoder = Seq2SeqAttentionDecoder2(len(data.tgt_vocab), embed_size,␣
 ↪num_hiddens, num_layers, dropout)
model = d2l.Seq2Seq(encoder, decoder, tgt_pad=data.tgt_vocab['<pad>'], lr=0.005)

trainer = d2l.Trainer(max_epochs=30, gradient_clip_val=1, num_gpus=1)
trainer.fit(model, data)
```

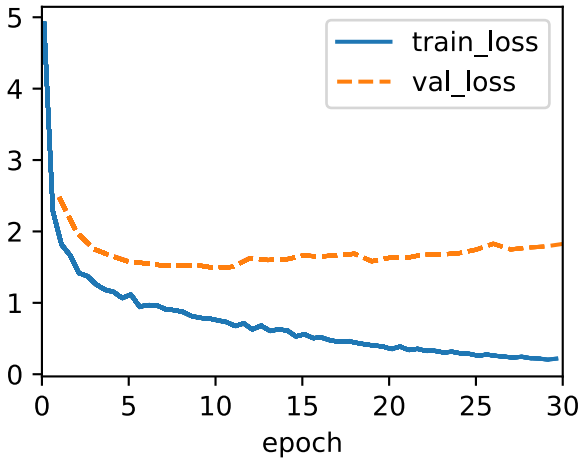


[80]:
```
engs = ['go .', 'i lost .', 'he\'s calm .', 'i\'m home .']
fras = ['va !', 'j\'ai perdu .', 'il est calme .', 'je suis chez moi .']
preds, _ = model.predict_step(
    data.build(engs, fras), d2l.try_gpu(), data.num_steps)
for en, fr, p in zip(engs, fras, preds):
    translation = []
    for token in data.tgt_vocab.to_tokens(p):
        if token == '<eos>':
            break
        translation.append(token)
    print(f'{en} => {translation}, bleu,'
          f'{bleu(" ".join(translation), fr, k=2):.3f}')
```

```
go . => ['va', '!'], bleu,1.000
i lost . => ["j'ai", 'perdu', '.'], bleu,1.000
he's calm . => ['<unk>', '.'], bleu,0.000
i'm home . => ['je', 'suis', 'chez', 'moi', '.'], bleu,1.000
```

2 Layers:

[81]:
```
data = d2l.MTFraEng(batch_size=128)
embed_size, num_hiddens, num_layers, dropout = 256, 256, 2, 0.2

encoder = d2l.Seq2SeqEncoder(len(data.src_vocab), embed_size, num_hiddens,␣
 ↪num_layers, dropout)
decoder = Seq2SeqAttentionDecoder2(len(data.tgt_vocab), embed_size,␣
 ↪num_hiddens, num_layers, dropout)
model = d2l.Seq2Seq(encoder, decoder, tgt_pad=data.tgt_vocab['<pad>'], lr=0.005)

trainer = d2l.Trainer(max_epochs=30, gradient_clip_val=1, num_gpus=1)
trainer.fit(model, data)
```

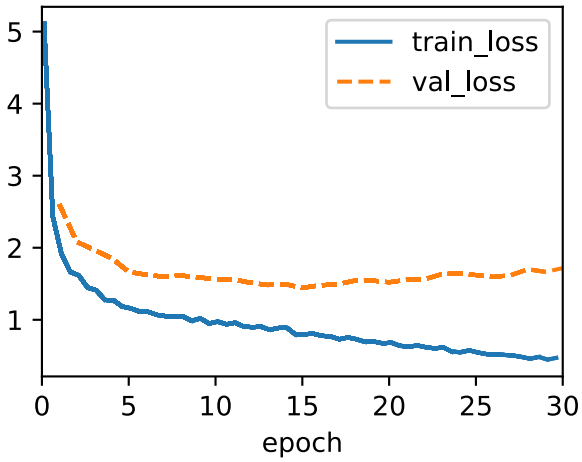


[82]:
```
engs = ['go .', 'i lost .', 'he\'s calm .', 'i\'m home .']
fras = ['va !', 'j\'ai perdu .', 'il est calme .', 'je suis chez moi .']
preds, _ = model.predict_step(
    data.build(engs, fras), d2l.try_gpu(), data.num_steps)
for en, fr, p in zip(engs, fras, preds):
    translation = []
    for token in data.tgt_vocab.to_tokens(p):
        if token == '<eos>':
            break
        translation.append(token)
    print(f'{en} => {translation}, bleu,'
          f'{bleu(" ".join(translation), fr, k=2):.3f}')
```

```
go . => ['va', '!'], bleu,1.000
i lost . => ["j'ai", 'gagné', '.'], bleu,0.000
he's calm . => ["j'ai", 'gagné', '.'], bleu,0.000
i'm home . => ['je', 'suis', 'gras', '.'], bleu,0.512
```

3 Layers:

[83]:
```
data = d2l.MTFraEng(batch_size=128)
embed_size, num_hiddens, num_layers, dropout = 256, 256, 3, 0.2

encoder = d2l.Seq2SeqEncoder(len(data.src_vocab), embed_size, num_hiddens,␣
 ↪num_layers, dropout)
decoder = Seq2SeqAttentionDecoder2(len(data.tgt_vocab), embed_size,␣
 ↪num_hiddens, num_layers, dropout)
model = d2l.Seq2Seq(encoder, decoder, tgt_pad=data.tgt_vocab['<pad>'], lr=0.005)

trainer = d2l.Trainer(max_epochs=30, gradient_clip_val=1, num_gpus=1)
trainer.fit(model, data)
```

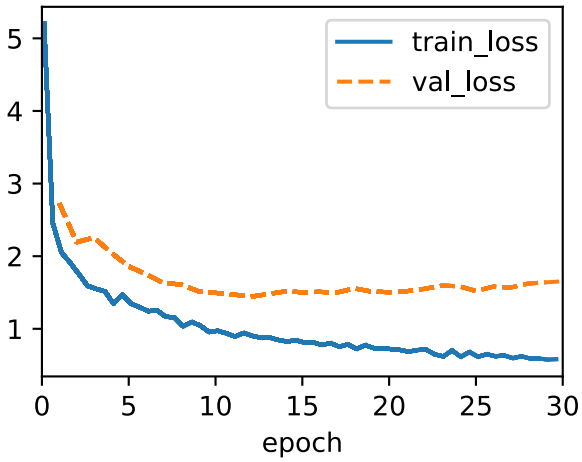


[84]:
```
engs = ['go .', 'i lost .', 'he\'s calm .', 'i\'m home .']
fras = ['va !', 'j\'ai perdu .', 'il est calme .', 'je suis chez moi .']
preds, _ = model.predict_step(
    data.build(engs, fras), d2l.try_gpu(), data.num_steps)
for en, fr, p in zip(engs, fras, preds):
    translation = []
    for token in data.tgt_vocab.to_tokens(p):
        if token == '<eos>':
            break
        translation.append(token)
    print(f'{en} => {translation}, bleu,'
          f'{bleu(" ".join(translation), fr, k=2):.3f}')
```

```
go . => ['<unk>', '.'], bleu,0.000
i lost . => ['j’ai', '<unk>', '.'], bleu,0.000
he's calm . => ['il', '<unk>', '<unk>', '.'], bleu,0.000
i'm home . => ['je', 'suis', '<unk>', '.'], bleu,0.512
```

4 Layers:

[85]:
```
data = d2l.MTFraEng(batch_size=128)
embed_size, num_hiddens, num_layers, dropout = 256, 256, 4, 0.2

encoder = d2l.Seq2SeqEncoder(len(data.src_vocab), embed_size, num_hiddens,␣
 ↪num_layers, dropout)
decoder = Seq2SeqAttentionDecoder2(len(data.tgt_vocab), embed_size,␣
 ↪num_hiddens, num_layers, dropout)
model = d2l.Seq2Seq(encoder, decoder, tgt_pad=data.tgt_vocab['<pad>'], lr=0.005)

trainer = d2l.Trainer(max_epochs=30, gradient_clip_val=1, num_gpus=1)
trainer.fit(model, data)
```



Prediction:

[86]:
```
engs = ['go .', 'i lost .', 'he\'s calm .', 'i\'m home .']
fras = ['va !', 'j\'ai perdu .', 'il est calme .', 'je suis chez moi .']
preds, _ = model.predict_step(
    data.build(engs, fras), d2l.try_gpu(), data.num_steps)
for en, fr, p in zip(engs, fras, preds):
    translation = []
    for token in data.tgt_vocab.to_tokens(p):
        if token == '<eos>':
            break
        translation.append(token)
    print(f'{en} => {translation}, bleu,'
          f'{bleu(" ".join(translation), fr, k=2):.3f}')
```

```
go . => ['<unk>', '!'], bleu,0.000
i lost . => ['je', 'suis', '<unk>', '.'], bleu,0.000
he's calm . => ['il', '<unk>', 'emporté', '.'], bleu,0.000
i'm home . => ['je', 'suis', '<unk>', '.'], bleu,0.512
```

Final conversion:

[ ]:
```
!apt-get install texlive texlive-xetex texlive-latex-extra pandoc
!pip install pypandoc
!apt-get update
!apt-get install inkscape
!add-apt-repository universe
!add-apt-repository -y ppa:inkscape.dev/stable
!apt-get update
!apt install inkscape
!jupyter nbconvert --to PDF "Homework5.ipynb"
```